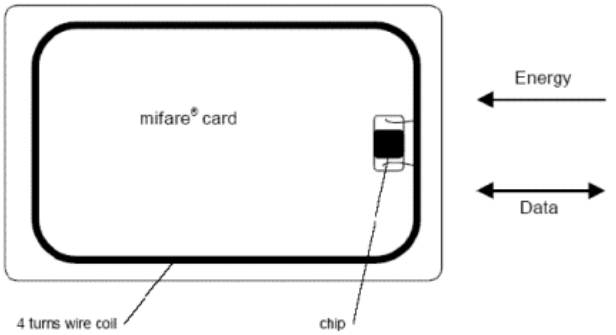
Breaking into Mifare Based Systems

- Bevezető - Balogh Gábor
- Dismantling Mifare Classic I. - Tánczos Zoltán
- Weaknesses and Exploits - Pósfay Márton
- Practical Attack on Mifare Classic - Germer Péter
- A Mifare Classic lehetőségei - Márton-Illés Krisztián

MIFARE

- Contactless smartcard
- 1994. NXP Semiconductors
- 500 millió kártya, 70%-os részesedés
- ISO/IEC 14443 Type A alapon
- Technológia a kártyában és az olvasóban
- Tömegközlekedés, beléptetés, jegyek

Típusok



- MIFARE Classic
  - Memóriával rendelkező kártya (1K/4K)
    - Hozzáférésvédelem szektoronként
    - Nettó méret 752/3440 byte
  - Saját authentikáló és titkosító protokoll
  - Crypto-1
  - Egyszerű felépítés → olcsó

Típusok

- MIFARE Ultralight
  - Classic-hoz hasonló, biztonsági elemek nélkül
  - 64 byte EEPROM, 10000 írási ciklus, 5 év
  - 7 byte Unique ID
  - Olcsó
    - Egyszer használatos jegyként használható
    - Vonaljegy, belépő (pl. 2006-os futball VB)

Típusok

- MIFARE ProX, SmartMX
  - ISO/IEC 14443-4-nek megfelelő smartcard
  - 1K/4K-s Classic kártyát emulál
  - 16K/72K memória, mikroprocesszor alapú → fel kell programozni
  - Kriptográfiai koprocesszor (3DES, AES, RSA)
  - Magas biztonsági szintű megoldásokban
  - SmartMX: EAL5+ minősítésű
    - Magas ellenállóság: reverse engineering, fault/glitch attack, power analysis

Típusok

- MIFARE DESfire
  - ProX/SmartMX-szel közös alap, van software
  - Classic-hoz képest több biztonsági megoldás
  - 3DES (+AES, CMAC – Cipher-based MAC)
- MIFARE Plus
  - Classic leváltására, visszafelé kompatibilis
  - 128 bit AES, random UID
  - Crytpo-1 hibáit védi, más jellegű támadásokat nem (brute-force, kriptoanalitikus)

Működés

- Kártya kiválasztása
  - Több is lehet a hatósugárban
- Belépés a szektorba
  - Kihívás-válasz alapú
  - Az olvasó bizonyítja, hogy jogosult
  - Szektoronként más kulcsok
  - Titkosított csatorna (Crypto-1) a kártya és az olvasó között

Működés

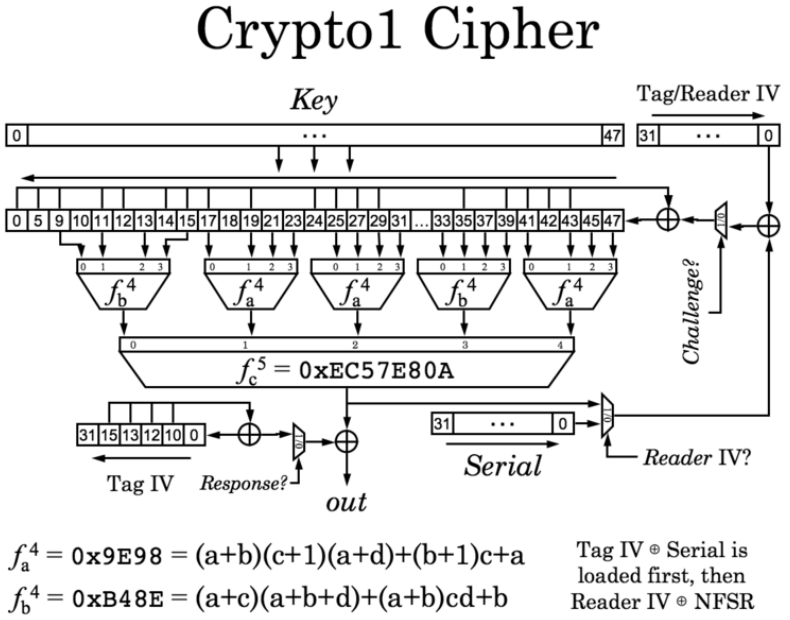
- Művelet elvégzése – hozzáférési szabályok szektoronként
  - Olvasás / írás / inkrementálás / dekrementálás
- Inkrementálás/dekrementálás
  - elektronikus pénztárca

Biztonság

- Szektorszintű hozzáférésvédelem
- ISO/IEC 14443 part 1, 2, 3 type A
  - Fizikai karakterisztika
  - Adóteljesítmény, és jelátvitel
  - Inicializálás és ütközésvédelem
  - (4 – átviteli protokoll)

Biztonság

- Crypto-1
  - Folyamkódoló
  - 48 bites kulcs
  - Saját eljárás
  - Sikeresen megtörték
    - Radboud University Nijmegen



Támadások

- Kriptográfiai algoritmus ellen
  - Sikeres
- Kimerítő kulcskeresés
  - 2^{48}×10ms = 89194 év (kiválasztás+belépés)
- Kulcs kinyerése
  - DPA (Differential Power Analysis)
- Az üzenetek nem védettek

Alkalmazás

- Oyster (London) – Classic 1K
  - 2003 július, London és környéke (busz, vasút, metró)
  - Vonaljegy/bérlet, zónarendszer, „pay as you go”
  - Olcsóbb, mint jeggyel (napszakonként is más lehet)
  - 2007 márc.: 10M kártya, utazások 80%-a (38M/hét)
  - Problémák:
    - 2004 jan.: PAYG bevezetése, bérletesek csak egyet utazhattak
    - 2005 márc.: software hiba, reggeli csúcsban leállt a rendszer
    - 2008 júl.: hibás software, 100000 kártya érvénytelenné vált
  - 2008 június: rendszer feltörése
    - Kártyaadatok megszerzése, kártya klónozás, 1 napig jó

Alkalmazás

- OV-Chipkaart (Hollandia) – Classic 4K
  - National Tariff System - busz, villamos, egyes vasutak
  - Rotterdam (2005), Amszterdam (2006, teljes 2008)
  - 2009-ben cserélik a papír alapú jegyeket
  - 3 változat
    - Eldobható vonaljegy – MIFARE Ultralight
    - Anonim kártya – többféle konstrukció
    - Személyhez kötött – teljes értékű, automatikus feltöltés
  - A teljes rendszer beindítása után a hamis jegyek kiszűrhetők lesznek
  - Az eldobható jegyek klónozhatók → ingyen utazás

Alkalmazás

- Touch and Go (Malajzia)
  - 1997-ben indították
  - Elsősorban autópályákon használható fizetésre
  - Kuala Lumpur-i tömegközlekedés, parkolás
  - Egyes helyeken fizetni is lehet vele
  - Pre-paid / Post-paid / Auto-reload
  - Hátrányok
    - A feltöltésnek költsége van
    - Használatával nem jár kedvezmény
    - Csak 35-50%-ban használják a fizetőkapuknál

Dismantling MIFARE Classic

Tánczos Zoltán Csaba

Flavio D. Garcia, Gerhard de Koning Gans, Ruben Muijrers, Peter van Rossum, Roel Verdult, Ronny Wichers
Schreur, Bart Jacobs: Dismantling MIFARE Classic (2008)

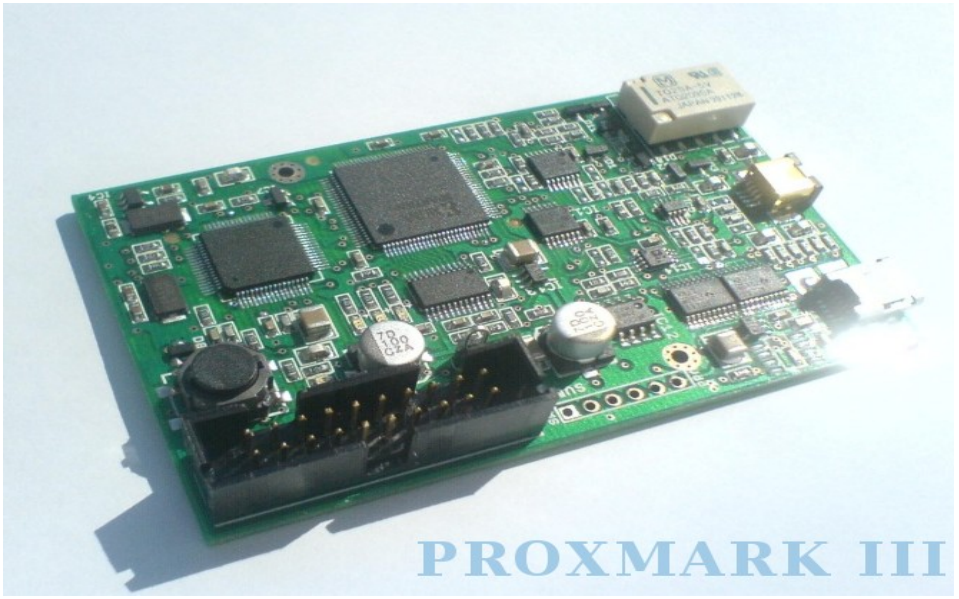
Dismantling MIFARE Classic[1]

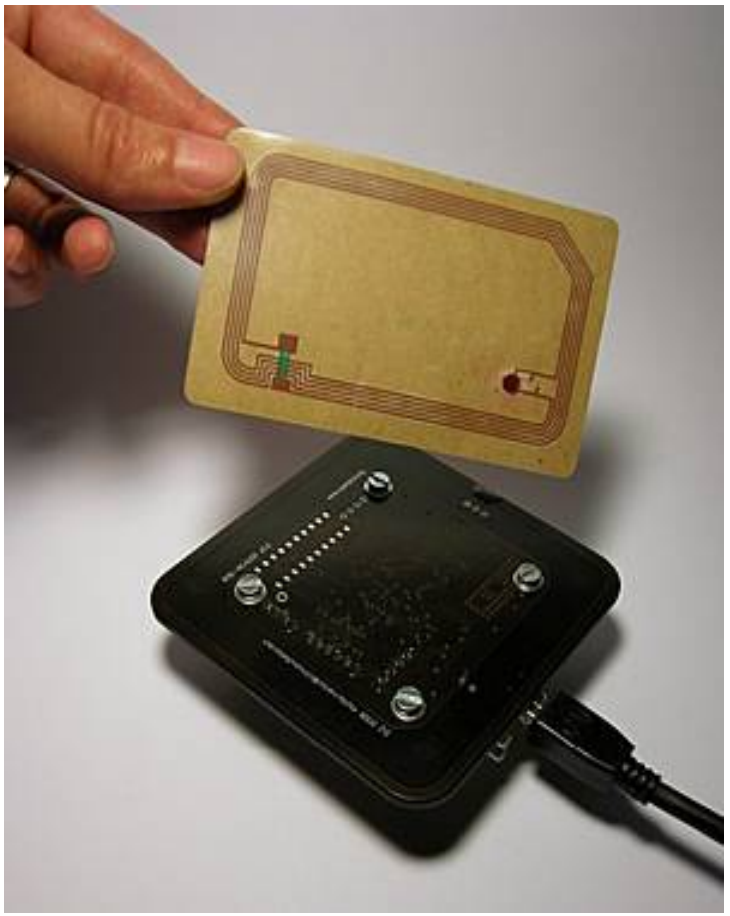
- Flavio D. Garcia et. al.:
  - *Reverse-engineering*-el visszafejtették az
    - Authentikációs protokollt
    - A szimmetrikus cipher-t
    - Az inicializációs mechanizmust
  - Bemutatnak két támadási lehetőséget:
    - A cipher inicializálásában felfedezett hiba
    - A CRYPTO1 cipher gyengesége miatt: <1 mp. alatt a titkos kulcs visszafejtése

"the security of this cipher is ... close to zero" – Wikipedia

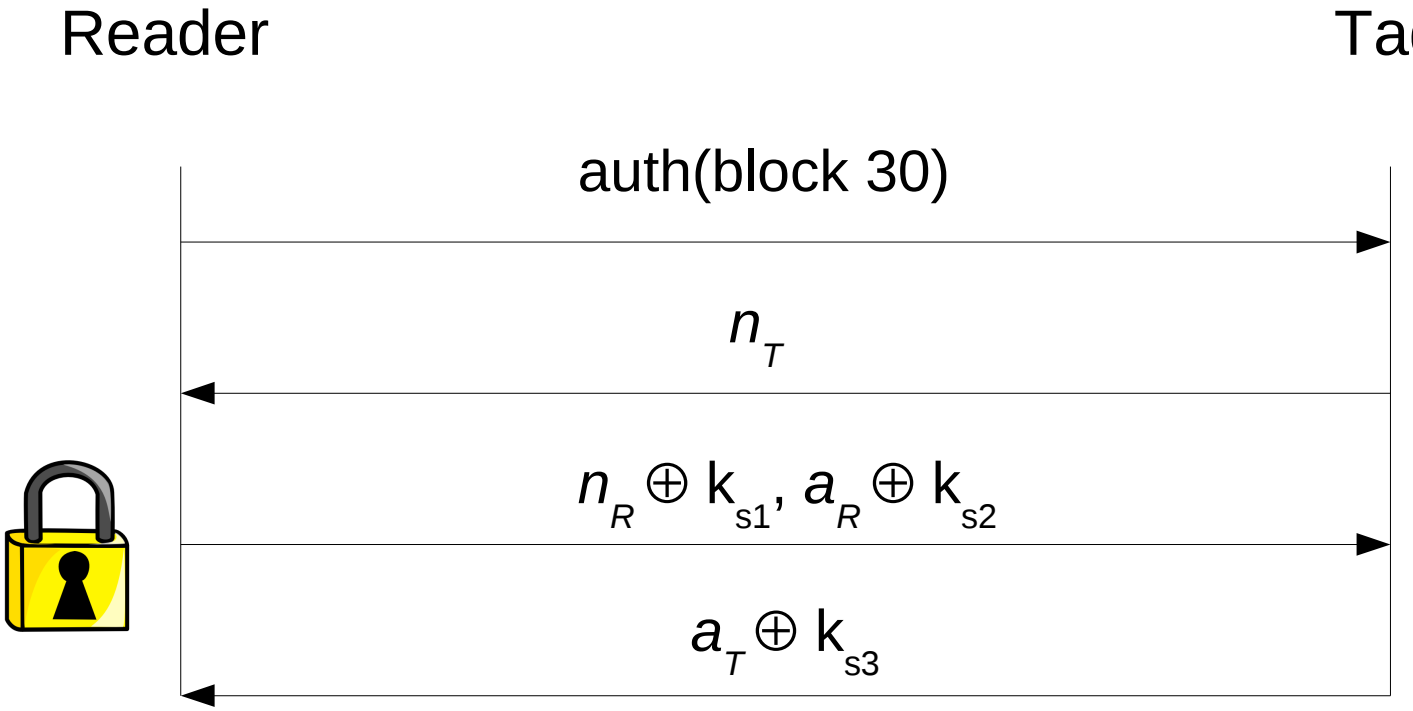
[1]: Flavio D. Garcia, Gerhard de Koning Gans, Ruben Muijrers, Peter van Rossum, Roel Verdult, Ronny Wichers Schreur, Bart Jacobs: Dismantling MIFARE Classic (2008)

- Felhasznált hardverek:
  - Ghost: tag emuláció (~40 eur)
  - ProxMark
  - Omnikey reader (~80 usd)
  - OpenPCD reader (~120 eur)





- Authentikációs protokoll:



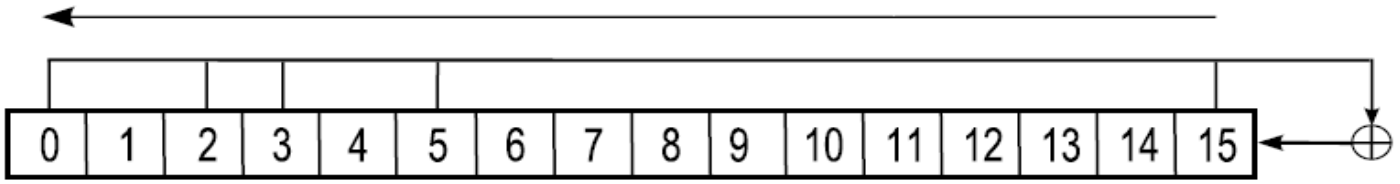
Ahol:

n_T, n_R: nonce, challenge 32-bit

a_T, a_R: response

k_{sx}: keystream

- A tag-ben levő PRNG:
  - Csak az indítás óta eltelt időtől függ → teljesen determinisztikus
  - 16 bites LFSR:
    - $g(x) = x^{16} + x^{14} + x^{13} + x^{11} + 1$



  - A nonce 32 bit-es, az LFSR 16:
    - n_T első fele meghatározza a másodikat

      → nem minden 32-bites szám lehet nonce

- n_T és uid változtatgatásával:

  uid: `0xc2a82df4` → $n_T \oplus$ uid = `0x803fed50`

  n_T: `0x4297c0a4` $n_R \oplus k_{s1}$ = `0x7ddb9b83`

  uid: `0x1dfbe033` → $n_T \oplus$ uid = `0x803fed50`

  n_T: `0x9dc40d63` $n_R \oplus k_{s1}$ = `0x7ddb9b83`

  - Ha $n_T \oplus$ uid konstans → $n_R \oplus k_{s1}$ is konstans
  - De a_T és a_R különbözik
  - Tipp: a kulcsfolyam ugyanaz, és a_T, a_R n_T-től függ

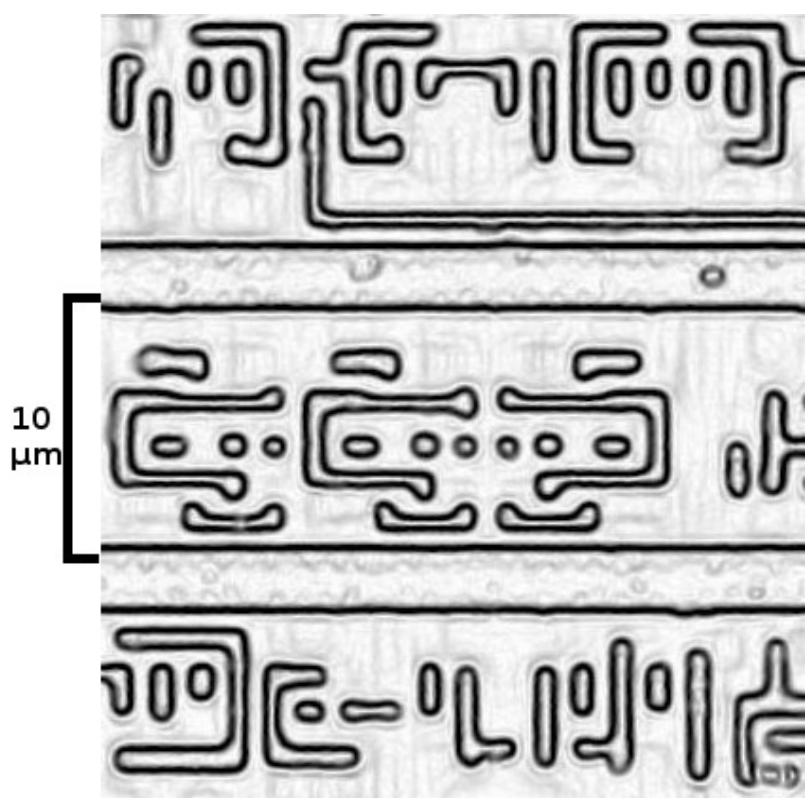
- $\mathrm{suc}(n_T)$: az LFSR által generált következő 32 bit
- $a_R \oplus a'_R = \mathrm{suc}^2(n_T \oplus n'_T) = \mathrm{suc}^2(n_T) \oplus \mathrm{suc}^2(n'_T)$
- $a_R = \mathrm{suc}^2(n_T)$ és $a'_R = \mathrm{suc}^2(n'_T)$
- $a_T = \mathrm{suc}^3(n_T)$ és $a'_T = \mathrm{suc}^3(n'_T)$

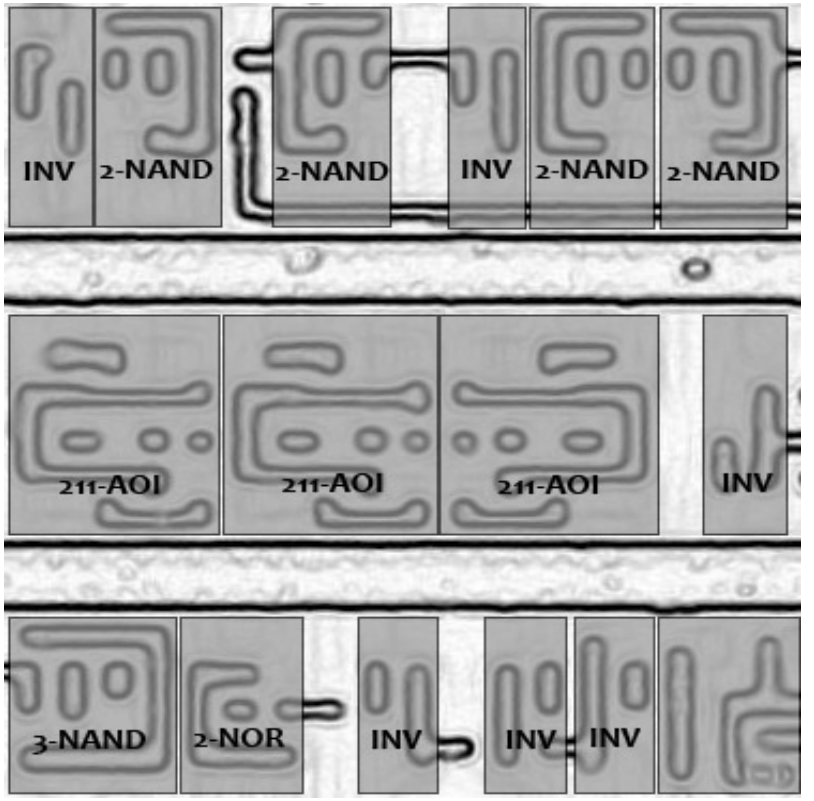
- Az authentikációs protokoll:

	Tag		Reader
0		anti-c(uid) $\longrightarrow$	
1		$\longleftarrow$ auth(block)	
2	picks n_T		
3		n_T $\longrightarrow$	
4	$\mathsf{ks}_1 \leftarrow \mathsf{cipher}(K, \mathsf{uid}, n_T)$		$\mathsf{ks}_1 \leftarrow \mathsf{cipher}(K, \mathsf{uid}, n_T)$
5			picks n_R
6			$\mathsf{ks}_2, \ldots \leftarrow \mathsf{cipher}(n_R)$
7		$\longleftarrow$ $n_R \oplus \mathsf{ks}_1, \mathsf{suc}^2(n_T) \oplus \mathsf{ks}_2$	
8	$\mathsf{ks}_2, \ldots \leftarrow \mathsf{cipher}(n_R)$		
9		$\mathsf{suc}^3(n_T) \oplus \mathsf{ks}_3$ $\longrightarrow$	

- A Keystream részleteinek kiderítése:
  - n_T és $\mathrm{suc}^2(n_T) \oplus k_{s2}$-ből:
    - $\mathrm{suc}^2(n_T)$ kiszámítása és XOR → k_{s2}
  - Ha a tag kihagyja a 9. lépést:
    - Reader → Tag: `halt` $\oplus k_{s3}$
    - `halt:` 0x500057cd (ISO14443A) → k_{s3}

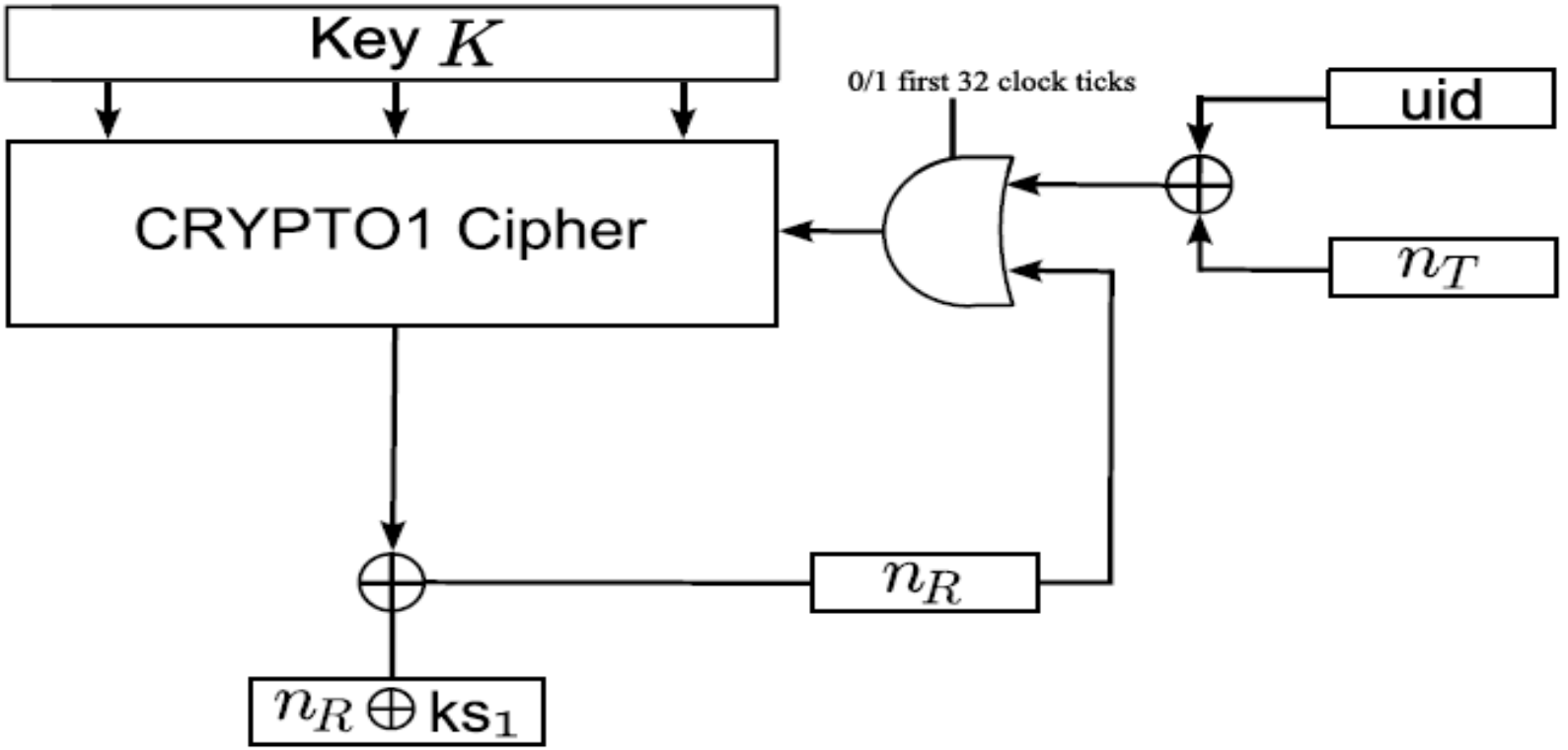
- `CRYPTO1` Cipher
  - 48-bites LFSR:
    - $g(x) = x^{48} + x^{43} + x^{39} + x^{38} + x^{36} + x^{34} + x^{33} + x^{31} + x^{29} + x^{24} + x^{23} + x^{21} + x^{19} + x^{13} + x^{9} + x^{7} + x^{6} + x^{5} + 1$
    - Ezt Nohl és Evans fejtette vissza





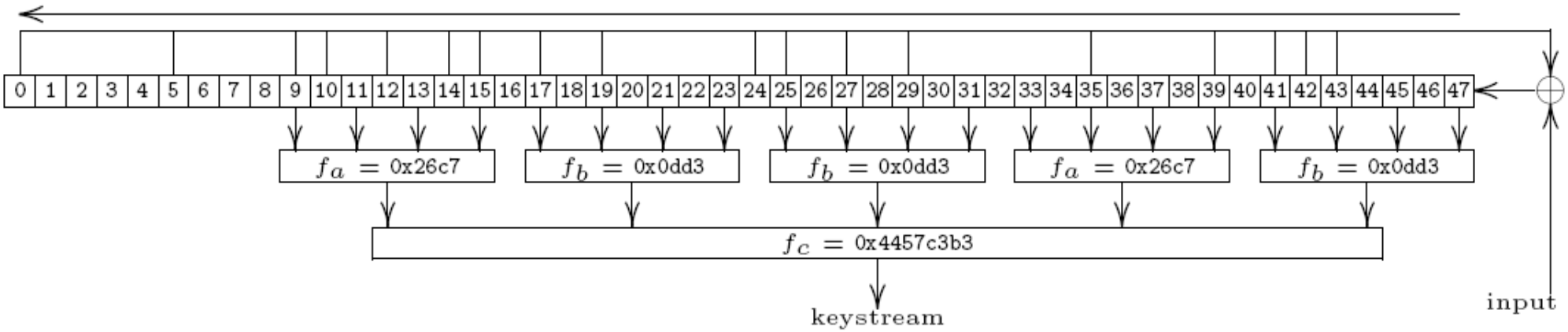
http://www.cs.virginia.edu/~evans/pubs/usenix08/mifare.html

- Inicializáció
  - LFSR kezdő állapota: K (secret key)
  - Az authentikáció során történik az init.



- *f* filter function
  - Kódoláshoz az LFSR kiválasztott bitjei átmennek *f* filter függvényeken

f_a = 0x26c7 f_b = 0x0dd3 f_c = 0x4457c3b3



- *Security through obscurity* elv
  - http://cryptolib.com/ciphers/crypto1/crypto1.c

    A CRYPTO1 cipher implementációja
  - http://code.google.com/p/crapto1/

    „Open implementations of attacks against the crypto1 cipher, as used in some RFID cards.”
  - Kerckhoff elv:

    „The security of a cryptographic system should not depend on the secrecy of the system itself, but only on the secrecy of the key.”

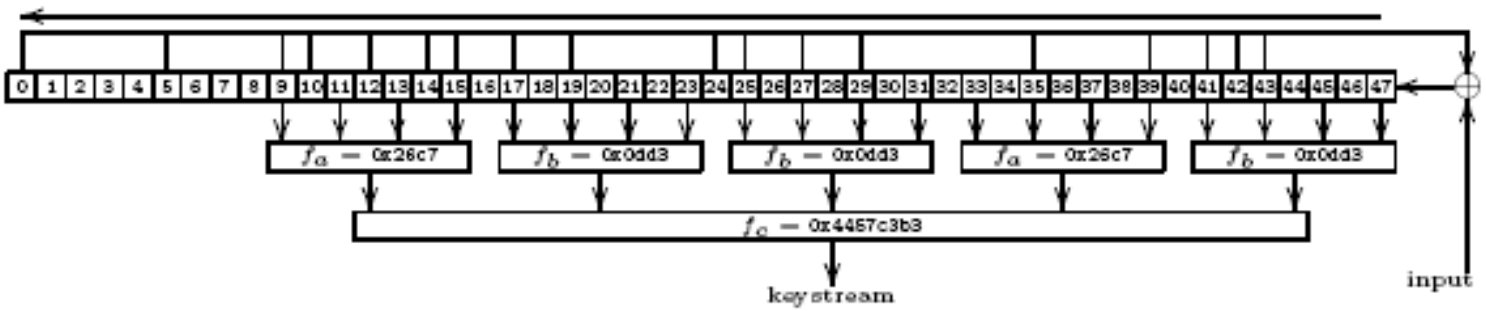
Weaknesses and Exploits

Pósfay Márton

Flavio D. Garcia, Gerhard de Koning Gans, Ruben Muijrers, Peter van Rossum, Roel Verdult, Ronny Wichers
Schreur, Bart Jacobs: Dismantling MIFARE Classic (2008)

Weaknesses and Exploits

- 4 tervezési hibát vizsgálunk
- 2 módszer
  - „A”
    - adatgyűjtés az olvasóról
    - f invertálás táblázattal
    - LFSR visszajátszás
  - „B”
    - f invertálás azonnal
    - LFSR visszajátszás



- Eredmény: MIFARE olvasó titkos kulcsa

„A” törés – adatgyűjtés, f invertálás

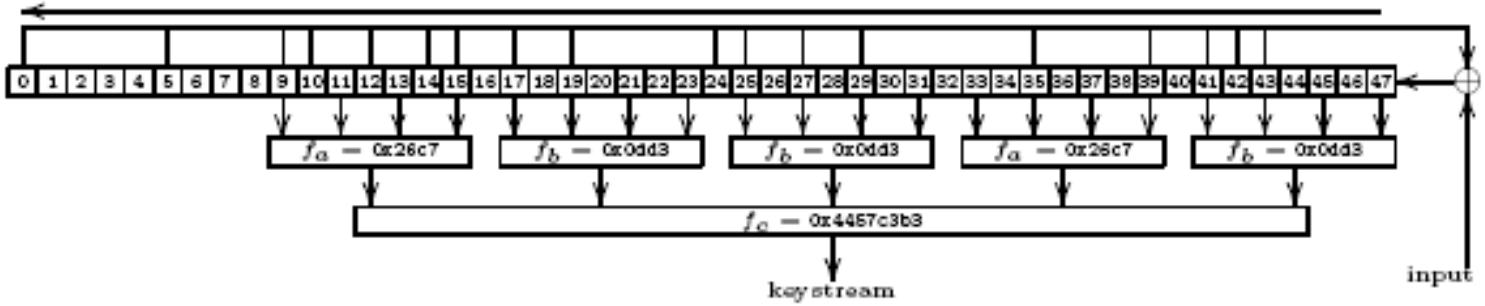
- Tag nonce -> LFSR állapot
- Táblázat létrhozása: LFSR|Ks
  - LFSR: lehetséges állapotok : `0x000WWWWWWWWW`
  - Ks: a kulcsfolyam első 64 bitje
  - újrafelhasználható
- Minden `0xXXX` értékre autentikációt kezdeményezünk
- n_T legyen `0x0000XXX0`

„A” törés – adatgyűjtés, f invertálás

- tag -> reader: n_T
- reader -> tag: n_R+ks_1, $suc^2(n_T)$+ks_2
- reader-> tag: halt+ks_3
- $suc^2(n_T)$ és halt ismert
- ks_2 és ks_3 kiszámítható
- `0xXXX` -> LFSR(`0xYYYYYYYY000Y`)
- n_R betáplálása -> LFSR(`0x000YZZZZZZZZ`)
- ks_2 és ks_3 segítségével kikeressük a táblázatunkból

„A” törés – LFSR visszajátszás

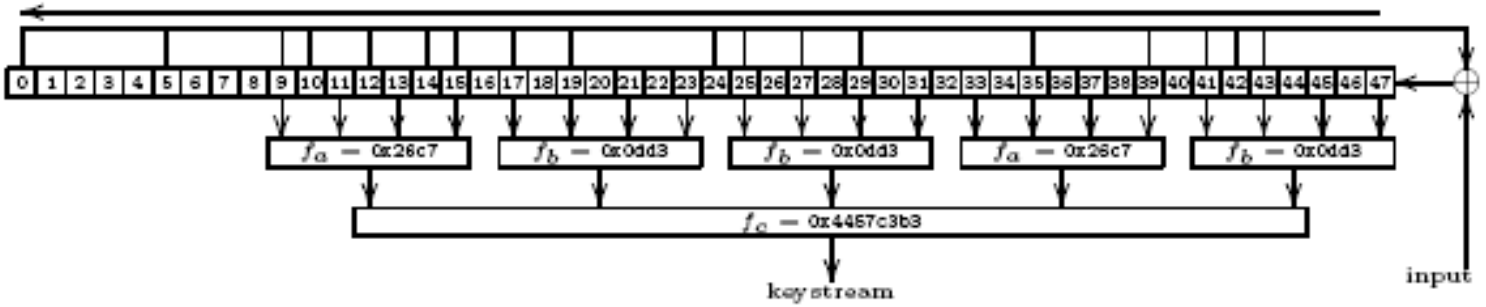
- k időpont: LFSR(r_k,r_{k+1},…,r_{k+47})
- Előző állapot (r_{k-1},r_k,…,r_{k+46}) számolható
  - $r_{k+48}=r_k+r_{k+5}+r_{k+9}+r_{k+10}+r_{k+12}+r_{k+14}+r_{k+15}+r_{k+17}+r_{k+19}+r_{k+24}+r_{k+27}+r_{k+29}+r_{k+35}+r_{k+41}+r_{k+42}+r_{k+43}+i$
  - i csak inicializáláskor használatos
  - Gyakorlatilag r_k-t keressük



A” törés – LFSR visszajátszás

- n_R-t nem tudjuk
- DE! f nem használja r_k-t LFSR-ből – ezt használjuk ki:
  - LFSR jobbra shift
  - bal bit legyen: r
  - f kiszámítása
  - $n_{R,31}$ visszaállítása
  - kiszámoljuk LFSR visszacsatolását -> r
- ezt ismételjük 31-szer -> LFSR állapot n_R előtt
- n_T és uid –t tudjuk - > LFSR n_T előtt : titkos kulcs

„B” törés – f invertálás



- f bemenetei: páratlan helyű értékek
- $b_0,b_1,\ldots,b_{n-1}$: kulcsfolyam
- Keresünk minden $r=r_0r_1\ldots r_{46+n}$ amire:

$$r_k+r_{k+5}+r_{k+9}+r_{k+10}+r_{k+12}+r_{k+14}+r_{k+15}+r_{k+17}+r_{k+19}+r_{k+24}+r_{k+25}+r_{k+27}+r_{k+29}+r_{k+35}+r_{k+39}+r_{k+41}+r_{k+42}+r_{k+43}+r_{k+48}=0$$

(k: 0…n-2)

$f(r_k\ldots r_{k+47}) = b_k$ (k: 0…n-1)

„B” törés – f invertálás

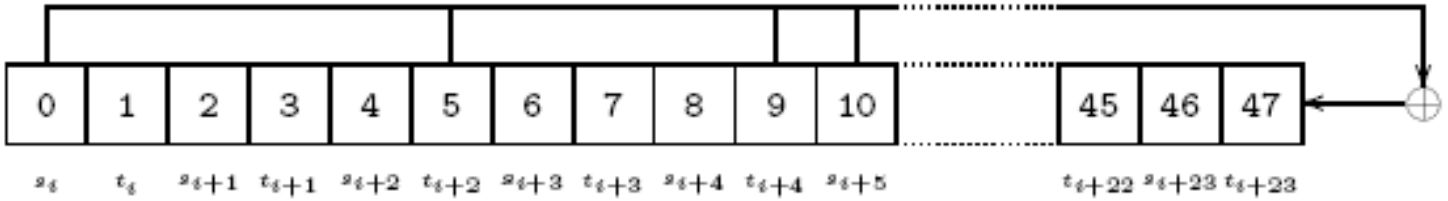
- Vesszük a kulcsfolyam első bitjét: b_0
- Legeneráljuk az összes 20 bites sorozatot, amire $f(s_0,s_1,\ldots,s_{19}) = b_0$
- 2^{19} ilyen van
- Minden ilyen sorozatra:
  - Az LFSR 9,11,…,47. bitjének vesszük
  - Shift balra kétszer
  - Második visszacsatolt bit: s_{20} (0/1)
  - LFSR többi bitje nem ismert -> nem ismerjük a visszacsatolást
  - Ellenőrizzük, hogy melyik jó: $f(s_1,s_2,\ldots,s_{20}) = b_2$
    - Egyik egyezik: kiegészítjük a listát
    - Mindkettő: megduplázzuk ezt a sort és kiegészítjük mindkettőt
    - Egyik sem: töröljük

„B” törés – f invertálás

- ¼-szer duplázás, ¼-szer törlés, ¼-szer megmarad -> ~2^{19} bejegyzés marad
- Megismételjük $b_4,b_6,\ldots,b_{n-1}$-re
- Ugyanezt elvégezzük a páratlan bitekre is
- Lesz két 2^{19} méretű táblánk:
  - $s_0s_1\ldots s_{19+n/2}$ -> $f(s_i,s_{i+1},\ldots,s_{i+19}) = b_{2i}$
  - $t_0t_1\ldots t_{19+n/2}$ ->$f(t_i,t_{i+1},\ldots,t_{i+19}) = b_{2i+1}$
- (brute force: 2^{48} helyett 2^{38})

„B” törés – f invertálás

- LFSR visszacsatolásának vizsgálata



- Veszünk mindkét táblánkból egy-egy bejegyzést
- $b_i^{1,s}=s_i+s_{i+5}+s_{i+6}+s_{i+7}+s_{i+12}+s_{i+21}+s_{i+24}$
- $b_i^{2,t}=t_{i+2}+t_{i+4}+t_{i+7}+t_{i+8}+t_{i+9}+t_{i+12}+t_{i+13}+t_{i+14}+t_{i+17}+t_{i+19}+t_{i+20}+t_{i+21}$
- $b_i^{1,s} = b_i^{2,t}$?
- Ezt szimmetrikusan fordítva is ($b'^{1,s}_i = b'^{2,t}_i$?)
- Az így kapott sorozatokat kiválogatjuk a táblázatból
- Mivel ezzel $r_9 r_{10} \ldots r_{46+n}$ –t kapjuk meg, még 9 lépéssel el kell tolni -> LFSR kezdőállapot

Practical Attack on Mifare Classic

Germer Péter

Gerhard de Koning Gans, Jaap-Henk Hoepman, and Flavio D. Garcia:
A Practical Attack on the MIFARE Classic (2008)

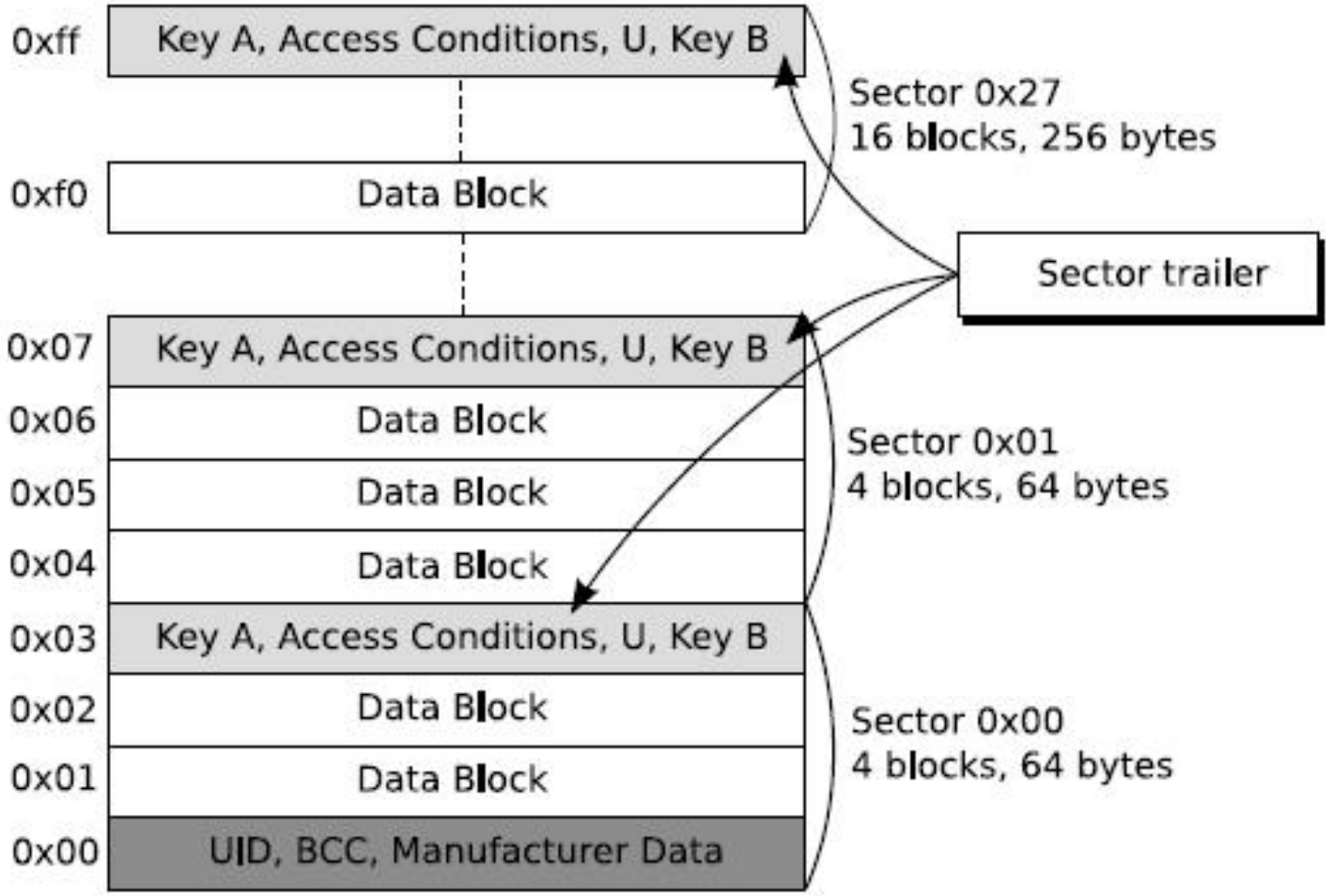
Practical attack on MIFARE Classic

- MIFARE Classic felépítése
- Parancsok felépítése
- MIFARE Classic gyengeségei, támadások
  - Kulcsfolyam visszafejtéses támadás
  - Kulcsfolyam feltérképezése
  - A Zero sector olvasása
  - Parancsok visszafejtése

MIFARE Classic felépítése

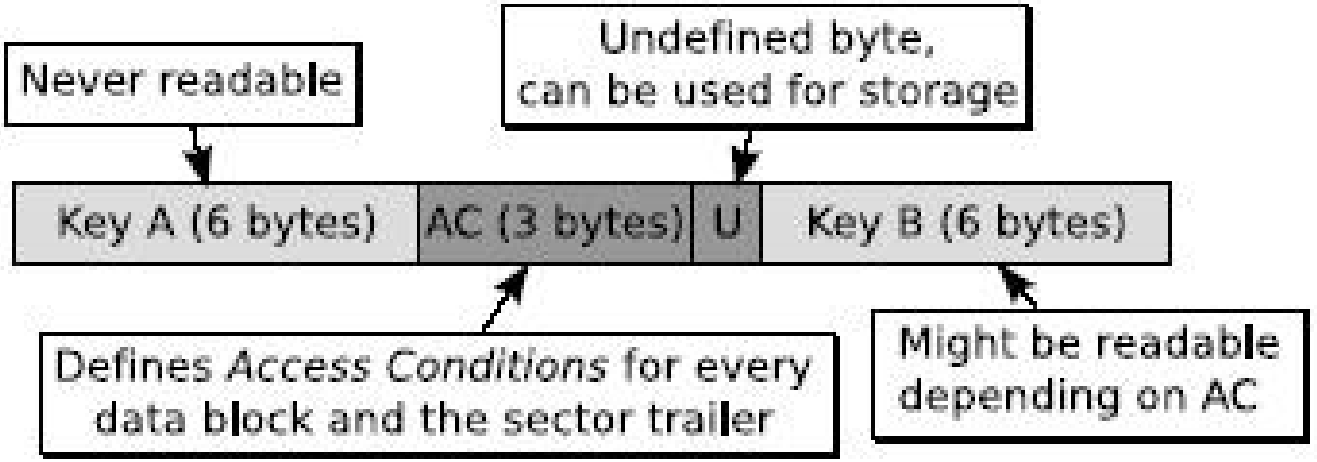
- A memória 16 byte-os blokkokra van osztva, amiket szektorokba csoportosítunk
- MIFARE Classic 1K: 16 szektor, mindegyikben 4 adat-blokk
- MIFARE Classic 4K: első 32 szektora 4 adatblokkot, a többi 8 szektor 16 blokkot tartalmaz
- Minden szektor utolsó blokkját *sector trailer*-nek nevezzük

MIFARE Classic felépítése



MIFARE Classic felépítése

- A *sector trailer* felépítése



Parancsok felépítése

- A parancsok kicsik, és mindig egy adott blokkra vonatkoznak
- A leolvasónak minden új szektor olvasásához azonosítania kell magát
- Parancsok:
  - *Read and Write*
  - *Decrement, Increment, Restore, Transfer* – csak *value* típusú blokkokra engedélyezett

MIFARE Classic gyengeségei

- A támadáshoz használt hardware a Proxmark III volt
- Segítségével óránként 600 000 véletlent tudtak kérni egy kártyától
- Egy-egy nonce óránként 4-szer ismétlődött
- A véletlent egy LFSR generálja, ami minden 9,14 microsecben léptet. Ez pontosan 1 bit periódus a kommunikációban
- Ezzel az információval egy korábban lehallgatott tranzakció megismételhető, amennyiben jó pillanatban sikerül megszólítani a kártyát

Kulcsfolyam visszafejtése

- Érdekes lehet, mert…
  - A támadás előállít plain texteket egy off-line brute force támadáshoz
  - A támadással részleteket tudhatunk meg a használt parancsokról
  - A kulcsfolyam használatával olvashatunk a kártyáról anélkül, hogy ismernénk a kulcsokat
  - A lehallgatott üzeneteket módosíthatjuk is, így akár a kártya tartalmát is módosíthatjuk (write parancs)

Kulcsfolyam visszafejtése

- A támadás menete:
  - A kommunikáció lehallgatása leolvasó és kártya között
  - Biztosítékra van szükség, hogy a kártya újra használni fogja a kulcsot (a leolvasót mi vezéreljük!)
  - Plain text módosítása, hogy a kártya olyan üzenetet kapjon, aminek ismerjük a tartalmát (pl. egy *read* üzenetben módosítjuk a blokk számot)
  - Az ismert plain text minden szegmensére kiszámoljuk a kulcsfolyam szegmenst
  - Felhasználjuk ezt a folyamot, hogy részben visszafejtsük az első pontban megfigyelt tranzakció további blokkjait
  - További kulcsfolyambitek visszafejtése a parancsok shiftelésével

Kulcsfolyam visszafejtése

- A titkosított C üzenet előállítása:

$$C = P + K$$

- Ekkor:

$$C1 = P1 + K$$

$$C2 = P2 + K$$

$$C1 + C2 = P1 + P2 + K + K = P1 + P2$$

- Így ha lehetőségünk van ugyanazt a kulcsfolyamot újra használni egy másik nyílt szövegre, akkor ismerjük az eredeti üzenetet, és a kulcsfolyam P-nyi szegmensét is

Kulcsfolyam feltérképezése

- Minden üzenet bitenként van titkosítva, a kulcsfolyam minden elküldött üzenet után léptetve van eggyel mindkét oldalon
- Nincs visszacsatolás, vagy szinkronizáció
- A parancsok változó hossza további nehézségeket jelentett a kulcsfolyam feltérképezésében
- A paritások is titkosítva vannak, méghozzá a kulcsfolyam azon bitjével, amelyet a következő adatbitek kódolására is használnak

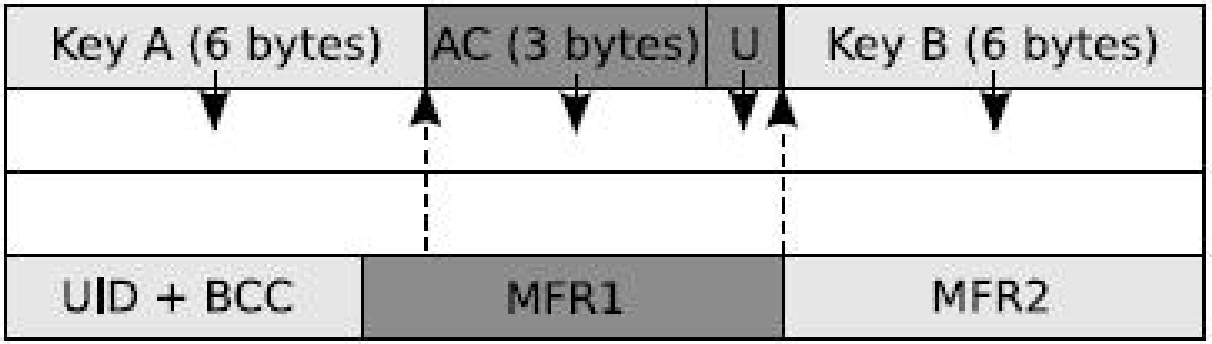
Kulcsfolyam feltérképezése

- Menete:
  - A visszafejtett kulcsfolyamból le kell venni azokat a biteket, amelyek paritásokra estek
  - A fennmaradó kulcsfolyam alkalmas új üzenetek titkosítására
  - Amennyiben paritást kell titkosítanunk, vegyük a következő kulcsbiteket az LFSR léptetése nélkül
  - Minden más esetben léptessük az LFSR-t.

Sector Zero olvasása

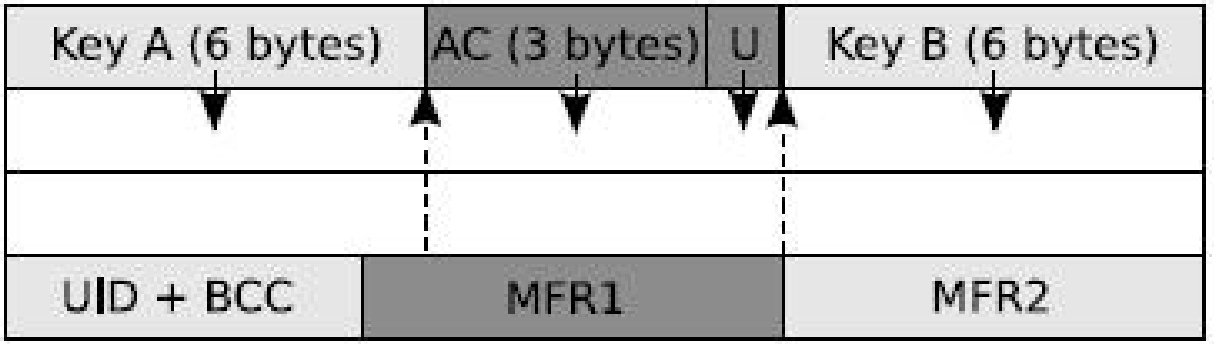
- Mindösszesen egy lehallgatott tranzakcióra van szükségünk
- Minden MIFARE Classic kártyának vannak ismert tartalmak a memóriájának ezen területén, amelyeket az NXP korábban publikált
- A lehallgatott kommunikációt visszajátszva csak az olvasni kívánt szektor és blokk módosításával próbáljuk a kulcsfolyamot visszafejteni

Sector Zero olvasása



- A sector trailer olvasásakor Key A értéke helyett logikai 0-kat kapunk, és a leggyakoribb konfiguráció esetében a Key B helyett is
- A 0. szektor 3. blokkját olvasva a kulcsfolyam első és utolsó 6 byte-ját kapjuk válaszul (C=0+K)
- Ugyanazt a tranzakciót újra lefuttatva a 0. szektor 0. blokkjára megkapjuk K 12 byte-jának ismeretében UID, BCC és MFR2 értékét, valamint MFR1 első byte-ját

Sector Zero olvasása



- A tapasztalatok szerint a Classic 1K és 4K kártyáknak MFR1 értéke ugyanaz, ami viszont lefedi az *Access Conditions* byteokat és az U byte-ot
- Az AC tárolási technikájának köszönhetően könnyen ellenőrizhetjük, hogy sikerült-e kiolvasnunk az AC byte-okat
- Az U byte értéke bármi lehet, de tapasztalatok szerint legtöbbször 00 vagy 96 értéket kap

Magasabb szektorok olvasása

- Magasabb szektorok olvasásakor nincs meg az MFR1 és MFR2 kínálta lehetőség, csupán a sector trailer-re támaszkodhatunk
- Key A miatt a kulcsfolyam első 6 byte-ja ismert, és a legtöbb esetben az utolsó 6 is
- Csupán az AC + U byte-ok által lefedett 4 byte-nyi kulcsfolyamról nincs információnk

Parancsok visszafejtése

- A kutatás idején nem tudták, hogy a szabadalmaztatott, titokban tartott parancsok elérhetőek az NXP honlapján egy demo firware-ben
- A feltérképezéshez egy üres adat-blokkokkal rendelkező, alapértelmezett kulcsokat használó kártyát használtak
- Minden parancs, amit a leolvasó küld, egy parancs byte-ból, paraméter byte-ból és 2 CRC byte-ból áll
- Feltételezték, hogy ismerik a parancsot, és ezt hozzá XOR-olták a titkosított üzenethez, ami a kulcsfolyamot adta eredményül
- Ellenőrzésképpen a kapott bitsorozatot XOR-olták egy parancshoz, aminek ismerték a válaszát egy korábban rögzített tranzakcióból

A Mifare Classic lehetőségei

Márton-Illés Krisztián

Wouter Teepe
w.teepe@cs.ru.nl
Radboud University Nijmegen
October 6, 2008

Bevezetés

A Mifare Classic-ról:

A Mifare Classic chipen használt kriptográfiai megoldások komoly hibákat tartalmaznak
“A Mifare Classic egy más, gyengébb biztonsági modellnek felel meg.”

Miért próbáljuk egyáltalán erősíteni?

Előfordulhatnak olyan esetek, amikor a migrálás egy másik kártyára lehetetlen, kivitelezhetetlen, vagy megfizethetetlen rövidtávon.

A tárgyalt támadások

Állapot-visszaállítás

- Mi az?
- Lehetséges ellenintézkedések.

Klónozás

- Mi az?
- Lehetséges ellenintézkedések.

Állapot-visszaállítás

Ebben a támadásban a támadó megszerez egy kártyát és saját előnyére manipulálja azt.

Kiolvassa a kártya állapotát.
Használja a kártyát, amíg az le nem merül.
Visszaállítja az eredeti állapotot, és újra használja a kártyát.

Ez **megelőzhető** H a kártya állapotát egy aláírás segítségével összekötjük egy szigorúan egyirányba változtatható számlálóval.

Szükséges hozzá:
Bizonyos adat-infrastruktúra
Kulcs-infrastruktúra
Allokációs-sémák a kártyán

Tárhelyet és számítási kapacitást áldozunk az erősebb biztonságért cserébe.

Követelmények

Kulcs-infrastruktúra

Minden eredeti állapotváltoztató olvasóban megbízunk és van egy publikus-kulcs infrastruktúra amely által minden eredeti állapotmódosító olvasó képes aláírni és minden eredeti olvasó képes ellenőrizni a más olvasók által generált aláírásokat.

Kártyaállapot aláírása

Minden tranzakció részeként, amely módosítja a kártya állapotát, egy visszavonhatatlan módosítást végzünk a kártyán. Az olvasó egy aláírást helyez el a kártya állapotán és az egyirányban módosítható területeten.

Kártyaállapot ellenőrzése

Minden olvasó ellenőrzi az aláírást, mielőtt tranzakcióba kezdene a kártyával.

Az egyirányú blokk

A szektor hozzáférésvezérlésének beállítása (access bits)

A trailer-t *restricted / frozen* módba állítjuk (access bitek innentől nem módosíthatóak)
A blokkot *restricted* módba llítju (Csak „egy irányban” módosítható)

Az egyirányú blokk egy ‘value block’, ami egy 32-bites integert tartalmaz, amelyet csak csökkenteni lehet.

Azonban:

Az elérhető tranzakciók száma nem $2^{32}-1$, csak nagyjából 5^{10}. Ennyiszer lehet írni az EEPROM-ot, mielőtt elhasználódik.

Mindazonáltal:

Az adatokra 10 év jótállás van
Ezt napi átlag 27 tranzakcióra elég, ha 10 évig minden nap elvégezzük őket
Nagyjából minden általános használatra elég ennyi

Problémák

Probléma: A Mifare Classic nem képes atomi szintű írási műveletekre.
Solution: Adatbázisokhoz és naplózó fájlrendszerekhez hasonló módon megoldható.

Fizikai állapot – A memória teljes tartalma.
Logikai állapot – A memória tartalmának egy korlátozott nézete. Az, amit jelent.

Számláló szektor – Egy visszafordíthatatlan számláló, és az írási tranzakciókért is ez felel.
Blokk allokációs tábla – Ez rendeli össze a fizikai és logikai állapotot.
Aláírás – ez jelöli az érvényes állapotokat, és az állapotok konzisztens voltát.

Állapotátmenetek

Állapotellenőrzés – A logikai állapotot kiolvassuk és ellenőrizzük. Ha a kártya nem érvényes, de visszaállítható állapotban van, visszaállítjuk. Ha az olvasó nem képes helyreállítani a kártyát, a tranzakciót megszakítjuk.
Érvényes állapot – A kártya egy konzisztens állapotban van. A számláló-szektor és az aláírás érvényesek. Néhány adatblokk használaton kívül van.
Tranzakció előkészítése – A nem használt blokkokba adatokat írunk úgy, hogyha a számlálót csökkentjük, a kártya érvényes állapotba kerüljön. Ebbe beletartozik egy új aláírás is az új állapoton.
Tranzakció commit-olása – A számlálót atomi szinten csökkentjük.
Új érvényes állapot – A kártya egy (új) konzisztens állapotban van. Néhány (más) adatblokk használaton kívül van.

Lehetséges eljárások:
Fifty-fifty
Partial fifty-fifty
Allocation table

Aláírás

Hash – 256 bit (2 blokk a Mifare Classic-on)

Aláíró algoritmus – RSA vs. DSA

- A 256 bites hash erősségének megfelelő erősség eléréséhez:
  - RSA: pq legalább 3248 bit (406 byte = 26 blokk)
  - DSA: A q (modulus) legalább 256 bites legyen, ami így 512 bites aláírást eredményez (64 byte = 4 blokk)

A DSA-t választjuk a kisebb tárigény miatt.

Részleges aláírás-ellenőrzés

Klónok megkülönböztetése az eredeti kártyáktól

A klón gazdaeszköz lehet:
Egy másik (üres) Mifare Classic kártya
Egy másik RFID kártya, amely “Mifare-kompatibilis”
Egy általános RFID emulátor, mint a Ghost vagy a Proxmark3

Side-channel információk:
Csupán a válaszidő elég, hogy különbséget tehessünk egy NXP Mifare Classic, egy Fudan, és egy NXP SmartMX (ami “Mifare-kompatibilis”) kártyák közt.
De: ez csak labor környezetben megbízható!

UID:
A Fudan egy eredeti Mifare Classic 4K-t imitál, de mivel eltérő a memóriafelosztása, ezt automatikusan detektálhatjuk.

Különbséget tudunk tenni egy eredeti, egy „kompatibilis” és egy Fudan kártya közt side-channel információk használata nélkül is. **Egy emulátor azonban még mindig becsaphat minket!**

(!) Az általános RFID emulátorokat programozhatjuk úgy (legalábbis elméletben), hogy a lehetőségeknek megfelelően minél jobban utánozzák a klónozott kártya side-channel karakterisztikáit, beleértve az eredeti Mifare Classic-ot is.

Klón-tesztelési módszerek

Kulcs tesztelés és adat tesztelés

A tesztek kulcsait soha nem használják, csakis a teszt alkalmával.
A klón maximum tippelhet, hogy jó-e a kulcs, és maximum random adatot tud küldeni a blokk tartalmaként, és remélni, hogy pont eltalálja a várt adatot (szinte 0 esély).
Azonban ha egy eredeti kártyának küldünk rossz kulcsot, akkor az *halted* állapotba kerül, és előről kell kezdeni a kommunikációt, viszont ekkor az olvasó már nem tudhatja, hogy ez ugyanaz a kártya-e.

On-line vs. Semi-offline tesztelés

On-line esetben az olvasó a központi adatbázisból lekér egy még nem használt tesztet a kártyához, azt végrehajtja, és kiderül az eredmény.
Semi-offline esetben greylistek, whitelistek és blacklistek használatával oldható meg a klónok detektálása. Ezesetben nem garantálható, hogy egy klón *SOHA* ne jusson át egy ellenőrzőponton.

Semi-offline tesztelés folyt.

Az utólagos logelemzések alapján gyanúsnak bizonyuló kártyák greylist-re kerülnek.
Ha egy olvasó egy iylen kártyával találkozik, elvégez rajta egy tesztet, amivel ott helyben meg tudja állapítani, hogy klónról van-e szó, vagy sem.
*Ha a kártya átmegy a teszten, az olvasó új ID-t ad neki, a régit pedig feketelistára teszi.
Minden kártyán van egy „hello-számláló” is.

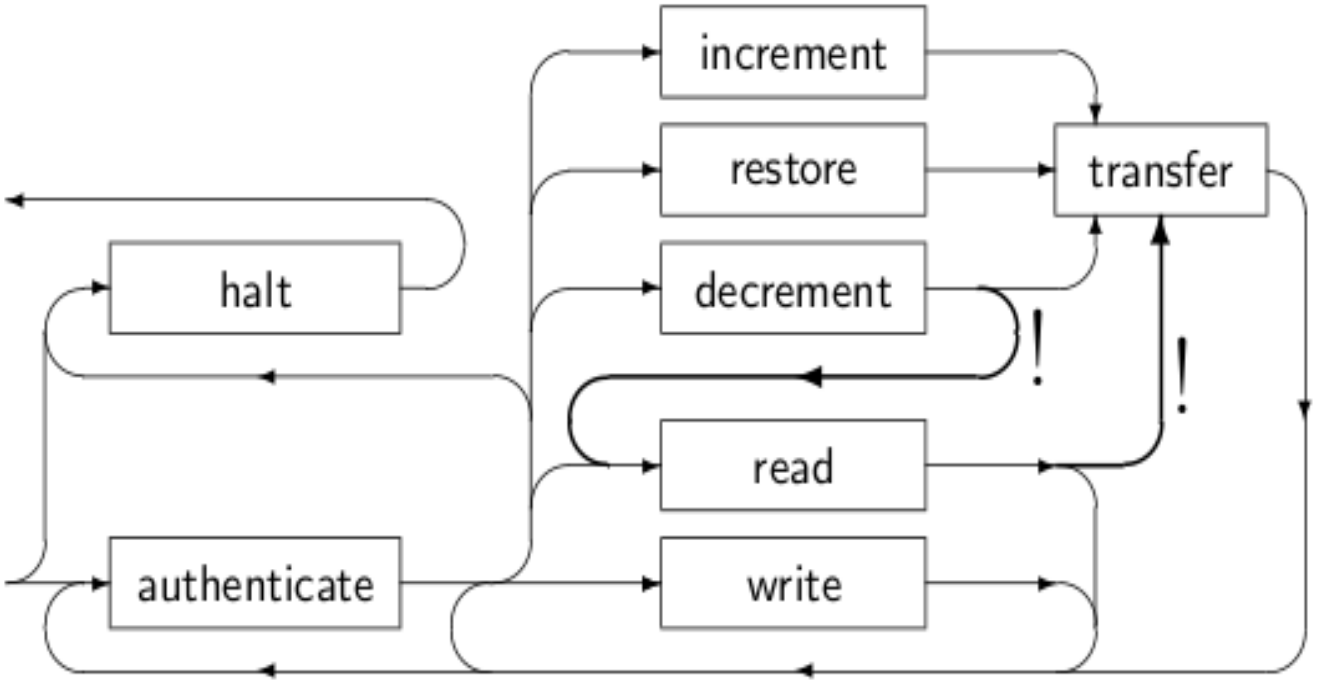
Egy ügyes támadás:
A támadó elkészít egy klónt, és ezt használva greylist-re tetteti az adott kártyát.
Ezután rögtön egy másik olvasóhoz megy, hogy megszerezze az adat-teszthez tartozó Crypto-1 kulcsot.
Ezzel a kulccsal visszamegy az áldozathoz, és kiolvassa az adatblokk tartalmát.
Ezután újra elmegy egy olvasóhoz, és a tesztre eredményül megadja a várt választ.
A klón új ID-t kap, az eredeti kártya pedig blacklist-re kerül. A támadó „ellopta” az áldozat kártyáját.

A fenti esetben fontos az időzítés. Ha az áldozat előbb ér egy másik kapuhoz, és ő kap ú ID-t, a támadónak nem volt szerencséje.

(!) Az eddigi, gyakorlatban használt megoldások mindegyike feketelistára teszi az eredeti kártyát is, jelentős bosszúságot okozva ezzel a jogos felhasználónak.

Megyjegyzések

Mindezektől függetlenül, mivel a Mifare Classic erősen épít a security by obscurity elképzelésre, valószínűleg számos olyan nem dokumentált funkciója van, amely miatt a fenti megoldások eredménytelennek bizonyulhatnak. Ilyenek például az alábbi állapotátmenetek is.



Köszönjük a figyelmet!

Kérdések?